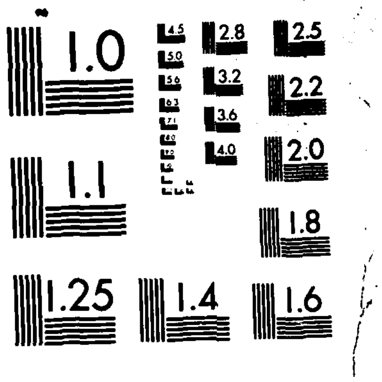
AD-A165 990 REASONING USING EXCLUSION AN EXTENSION OF CLAUSAL FORM (U) ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE J D TENENBERG JAN 85 TR-147 N00014-80-C-0197 1/1

UNCLASSIFIED F/G 9/4 NL


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

NR 4331425-01

80-C-0197

12

AD-A165 990

DTIC FILE COPY

Reasoning Using Exclusion: An Extension of Clausal Form

Josh D. Tenenberg
Dept. of Computer Science
University of Rochester
Rochester, NY 14627

TR 147
January 1985

rochester

Department of Computer Science
University of Rochester
Rochester, New York 14627

86 3 17 176

Reasoning Using Exclusion: An Extension of Clausal Form

Josh D. Tenenberg
Dept. of Computer Science
University of Rochester
Rochester, NY 14627

TR 147
January 1985

Abstract

In formalizing knowlege for common sense reasoning, one often needs to partition some domain. An instance of this from the block's world is the statement "All blocks are one of held, on the table, or on another block." Although we can write such an axiom in predicate calculus using the standard logical connectives, or in clause form as input to a resolution theorem prover, such representations are highly space inefficient. A generalized clause form is presented that allows for the compact representation of arbitrary partitions, along with a set of corresponding inference rules whose soundness and refutation completeness are proven. Additionally, an implementation of a subset of these rules is described that demonstrates their utility with certain kinds of common sense rule bases.

This work was supported in part by an Office of Naval Research Dept. of the Navy grant number N00014-80-C-0197 and the Rome Airforce Development Center (Syracuse University) number AF30602-81-C-0169.

1. Introduction

Reasoning about taxonomies has received attention, most notably within graph-structured representations. Shapiro (1979) introduces the AND-OR operator, which when applied to a set of *n* literals indicates that between *i* and *j* of them, inclusive, are true. This has a rather simple representation within a semantic network, although the details of its use and properties are embedded within the graph search procedures. Hendrix (1979) develops a partitioned associative network through the judicious use of subset and disjoint arcs. Stickel (1982) discusses taxonomic reasoning by using a search through a connection graph built from a set of assertions. The work described in this paper contrasts with those cited above in combining its use of purely declarative knowledge, its generality in allowing one to reason about non-ISA as well as ISA partitions, and its simple inference rules, which can be added to an existing declarative theorem prover to enable more efficient taxonomic reasoning.

2. Representation

Two examples that are typical of the relations that we wish to capture in commonsense reasoning are "All liquids either rest on a surface, rest in a container, flow along a surface, flow within a conduit, or fall in free space" (Hayes 1978) and "For any two times t_1 and t_2, either $t_1<t_2$, $t_1=t_2$, or $t_1>t_2$". These kinds of partitions are more general than ISA hierarchies. We can represent the latter statement using predicate calculus in conjunctive normal form (CNF) by

$$(t_1<t_2 \vee t_1=t_2 \vee t_1>t_2) \wedge [\sim(t_1<t_2) \vee \sim(t_1=t_2)] \wedge$$
$$[\sim(t_1=t_2) \vee \sim(t_1>t_2)] \wedge [\sim(t_1<t_2) \vee \sim(t_1>t_2)],$$

where the first disjunction indicates that one of the conditions must hold, and the next three indicate that not more than one can hold. Unfortunately, as the number of partitions grows linearly, the CNF formula grows quadratically, as all partitions must be pairwise compared. We will introduce an *n*-ary exclusive-or operator, X, a syntactic variant of a connective used in (Hayes 1978) to capture this relationship among the partitions, and will write the above axiom

$$X(t_1<t_2,\ t_1=t_2,\ t_1>t_2).$$

The truth table for X is true just when exactly one of the literals inside is true, false otherwise. Using this notation, the expression of axioms like the above requires a linear amount of space relative to the number of atoms, rather than the quadratic amount previously needed. This notation can be generalized, and any expression of the form

$$X(B_1, B_2, \dots, B_n)$$

where the B_i are literals will be called an *X-bundle*. The equivalent CNF formula, which will be called the *Expansion-Set* of $X(B_1, B_2, \dots, B_n)$ is the conjunction of all members of the set

$$\{B_1 \vee B_2 \vee \dots \vee B_n\} \cup \{\sim B_i \vee \sim B_j \mid 1 \le i < j \le n\}.$$

The *degree* of an X-bundle is the number of literals within it, so that the above X-bundle has degree *n*. Note that an X-bundle of degree one is equivalent to the literal it contains, and will usually be identified with that literal. An X-bundle of degree two is equivalent to the standard binary XOR connective, that is

$$X(B_1, B_2) \equiv B_1 \oplus B_2.$$

An expression having the form

$$A_1 \vee A_2 \vee \ldots \vee A_n$$

where each of the A_i are X-bundles, rather than simple literals, will be called an *X-clause*. This form reduces to regular clausal form when the degree of each A_i is one. If at most a single A_i has degree greater than one, we will call this a *singular* X-clause. For example, the clauses

$$Y \vee Z \text{ and } W \vee Y \vee X(B_1, B_2, B_3)$$

are singular X-clauses, while

$$X(B_1, B_2, B_3) \vee X(C_1, C_2)$$

is not. A conjunction of X-clauses is in *exclusive normal form* (XNF), and as usual, we will think of this as a set. Any set of clauses in CNF is therefore trivially in XNF. Converting the X-clause $A_1 \vee A_2 \vee \ldots \vee A_n$ where each A_i is an X-bundle, to a CNF clause produces

$$\{a_1 \vee a_2 \vee \ldots \vee a_n \mid a_i \in \text{Expansion-Set}(A_i)\},$$

where Expansion-Set is as defined above. The space savings to represent a set of assertions as compared to CNF can thus theoretically be an exponential, based on the number of X-bundles, although most practical cases will involve only singular X-clauses where the savings will be quadratic.

This syntax enables us to make simple statements such as the Blocks one given earlier, but it allows additionally for the expression of more complex statements, such as "All dogs are either short and slow, tall and quick, or of medium height, fat, and grumpy" with

$$Dog(y) \Rightarrow X(A(y), B(y), C(y))$$

if we furthur define within this axiomatic dog theory the predicates A, B, and C to be the described conditions. As another example, we have (taken loosely) from Allen's [1] temporal logic the following

$$\begin{aligned}&TimeInterval(x) \wedge TimeInterval(y) \Rightarrow\\ &\quad X(Before(x,y),\ Before(y,x),\ Meets(x,y),\ Meets(y,x),\\ &\qquad Overlaps(x,y),\ Overlaps(y,x),\ During(x,y),\ During(y,x),\\ &\qquad Starts(x,y),\ Starts(y,x),\ Finishes(x,y),\ Finishes(y,x),\\ &\qquad Equal(x,y)).\end{aligned}$$

{Note from the above that free variables are taken to be universally quantified, and that small letters from the end of the alphabet are meant as variables. Examples throughout will follow these rules, with the addition that small letters from the beginning of the alphabet are constants.} This X-clause would require 79 clauses containing a total of 327 atoms if it were expressed in CNF. This representaion will be coupled with inference rules that will allow for shorter proofs than using standard resolution. Shorter proofs are highly desirable since proof trees typically fan out at an exponential rate as a function of their depth. The combination, then, of fewer clauses in the data base, and shorter proofs, results in more efficient reasoning with taxonomic information.

3. Inference Rules

The primary task remaining is to develop a set of inference rules that meet the following conditions. First, they must have the formal property of soundness. We do not wish to draw any conclusions that we would not be able to make were we dealing with the equivalent CNF clauses. Second, we would like our rules to be sufficiently constrained so that the number of inferred clauses does not overwhelm our data base. Third, we would like our rules to be sufficiently powerful to be formally complete. That is, using X-clauses and the new inference rules we should be able to build a proof for each clause that can be proven from the equivalent initial clause set in CNF using a complete proof method, such as binary resolution with factoring. There is a natural conflict between points two and three. If we are to have completeness, then we must deal at some point with an exponentially growing set of inferred clauses. Our goal will be to slow this growth initially, so that for many proofs that are too difficult (require too much time or space to be practically solved) using standard CNF representations we will arrive at a refutation before the data base has become unmanageable. We will describe a set of inference rules that is sound and complete, with proofs of this provided in the appendices. We will then provide a subset of these rules that has better computational properties for the type of problems that we wish to solve, but which is incomplete.

Our initial set of rules was developed from looking at all of the ways in which an atom can occur twice in one parent clause, as in factoring, or in two parent clauses, as in binary resolution (Robinson 1965). Essentially, these inference rules involve unbundling the X-clauses slowly, peeling off the constituent X-bundles one at a time. Although all rules below are stated for propositions, the same rules apply to predicates, where the matching is between unifiable atoms. Additionally, in the rules that follow, α will be used to indicate a disjunction of some number of X-bundles, and will mean "whatever else follows", thus allowing us to focus on a single X-bundle where the unification will take place in a clause while disregarding the other X-bundles in this clause. Likewise, we will use a boldfaced, underlined capital letter to represent the remaining literals in an X-bundle.

The first rule generalizes binary resolution, and will be called X-resolution. This essentially states that resolution "works" with X-bundles in the same way that regular resolution works with literals - the remaining literals in the X-bundles of the unified literals get joined together in a single X-bundle in the child. In its general form, from the parent clauses

$X(B, \underline{C}) \vee \alpha_1$ and $X(\sim B, \underline{E}) \vee \alpha_2$

we can infer

$X(\underline{C}, \underline{E}) \vee \alpha_1 \vee \alpha_2$.

So, using this rule, from the X-clauses

X(Arch(B), Block(B), Wall(B)) and

X(~Arch(B), Support(B), Tower(B))

we can conclude

X(Block(B), Wall(B), Support(B), Tower(B)).

If the unified literals are within X-bundles of degree one, that is, the parents in the general case are the clauses

$B \vee \alpha_1$ and $\sim B \vee \alpha_2$

then this rule reduces to regular resolution. So for example, the clauses

Block(A) and

~Block(y) ∨ X(OnTable(y), Held(y), OnOther(y)).

entail

X(OnTable(A), Held(A), OnOther(A)).

We can also unify between literals from X-bundles of degree one, and literals from X-bundles of higher degree. The result is that all of the X-bundles from the parents get disjoined together, with the unified literal removed from each of these bundles. Thus, from

X(Wall(y), Block(y), Arch(y)) and

~Wall(z) ∨ Stationary(z)

we can infer

Stationary(y) ∨ X(Block(y), Arch(y)).

The second rule, called exclusion, unifies atoms that are the same, rather than complementary. It is justified from a very simple fact - if we know that exactly one of some set of conditions holds, and we also know that one of these conditions does in fact hold, then we can infer that all of the rest do not hold. For example, from the X-clauses

OnTable(A) and

X(OnTable(A), Held(A), OnOther(A))

we can infer both

~Held(A) and ~OnOther(A).

The general form of this rule states that from the X-clause parents

$X(B, \underline{C}_k) \vee \alpha_1$ and $X(B, \underline{E}_n) \vee \alpha_2$

(where the subscripts on C and E indicate the number of remaining literals in their respective X-bundles) we can infer any member from the set of clauses

$$\{\sim E_i \vee X(\underline{C}_k) \vee \alpha_1 \vee \alpha_2 \mid 1 \leq i \leq n\} \cup$$
$$\{\sim C_j \vee X(\underline{E}_n) \vee \alpha_1 \vee \alpha_2 \mid 1 \leq j \leq k\}.$$

Using this rule in a more complex case, from the X-clauses,

X(Held(y), OnTable(y), OnOther(y)) $\vee$ ~Block(y) and
X(Held(B), WeighsATon(B))

when we unify on Held, we can infer the following three X-clauses

~WeighsATon(B) $\vee$ X(OnTable(B), OnOther(B)) $\vee$ ~Block(B),
~OnTable(B) $\vee$ WeighsATon(B) $\vee$ ~Block(B), and
~OnOther(B) $\vee$ WeighsATon(B) $\vee$ ~Block(B).

To guarantee completeness in a resolution theorem prover, factoring is necessary to make inferences within a single parent clause when two literals unify. This allows us to prove P(a) from the clause P(a) $\vee$ P(x). There is an analogous situation with X-clauses, although it is more complex, so that in order to guarantee completeness we will need an inference rule for each of the ways in which atoms can unify within a single X-clause (1) the same atom within the same X-bundle -- X(A,A,$\underline{C}$), (2) the same atom in different X-bundles -- X(A,$\underline{C}$) $\vee$ X(A,$\underline{E}$), (3) complementary atoms in the same X-bundle -- X(A,~A,$\underline{C}$), and (4) complementary atoms in different X-bundles -- X(A,$\underline{C}$) $\vee$ X(~A,$\underline{E}$). These rules happen to be the worst computationally, but they also cover the least likely cases. These rules are listed in the appendices. A restricted version with better computational properties will be given later.

4. Restricted Inference Rules

A prototype theorem prover was built to implement these rules. It soon became clear that adding entire sets of clauses to the data base from a single inference is costly, due to the fact that each of these new clauses must be checked as possible parents in future deductions, although few if any of these clauses might be used in a proof. When the application of an X-rule results in more than one child, this indicates that some X-bundle is being decomposed, effectively negating the benefit of the representation. This problem was handled by restricting the clauses in the data base to be only singluar X-clauses (clauses having at most one X-bundle of degree greater than one), and by adding large sets of clauses from a single inference only when each child is a unit clause. Let us take as an example the following clause set.

1) ~A $\vee$ ~B_1
2) ~A $\vee$ X(D_1, D_2, D_3)
3) B_1
4) B_n
5) ~C
6) A $\vee$ C $\vee$ X(B_1, ..., B_n)

All of these clauses are singular. With the unrestricted rules, we can make inferences with 1 and 3, 1 and 6, 3 and 6, 4 and 6, 2 and 6 and 5 and 6. With the restrictions, we are disallowing 2 and 6, since the child clause $C \vee X(B_1, ..., B_n) \vee X(D_1, D_2, D_3)$ is non-singular. Additionally, the inference with 3 and 6 is disallowed, as the children, all members of the set $\{A \vee C \vee \sim B_i \mid 2 \leq i \leq n\}$, are non-unit clauses. Likewise for the inference with 4 and 6. What this restriction amounts to, is that before we will "infer into" an X-bundle, we will "resolve away" the other literals in the X-clause (that is, the X-bundles of degree one). Figure 1 illustrates the entire search space given the set of restricted inferences on this clause set. We will first resolve 1 and 3 to get ~A, and then resolve ~A with 6, resolving the child with 5 to get the clause $X(B_1, ..., B_n)$. We can now resolve this with either 3 or 4, since the set that is inferred will all be unit clauses - $\{\sim B_i\}$. Had we initially allowed the inference with 3 and 6 to be made, then each of the clauses 1, 2, 4, and 5 will succeed as a mate for an inference with each of the children from the inferred set, although few of these inferences would be useful in a proof. If we continued by applying all possible inference rules to all of these clauses and their descendants, even requiring that one of the parents be a unit clause, our search will requrie a number of inferences which is a linear function of n, the number of literals in the large X-bundle from X-clause 6. Had we written this clause set in CNF, it would have required on the order of $n^2/2$ clauses, with greater than n^2 possible resolutions in its entire search space of unit resolutions. If, however, we use the X-rules with the restriction mentioned above that we describe, then the entire search space of all possible inferences of these clauses and their descendents amounts to only 9 inferences as can be seen from figure 1.

It is clear that these restrictions reduce the search significantly, but is it too constrained? Based upon typical common sense rule bases it does not appear to be so. First, requiring initial data bases to contain singular X-clauses is not a large constraint, since most taxonomic information is of the form "An x is one of a, b, c, or d", which is expressable as a singular X-clause. Second, it is not clear if completeness is sacrificed from these restrictions, as no examples where this has been the case have been found. The restrictions merely serve to order the clauses so that the high density information of the X-clause can be extracted in an orderly fashion.

A selection strategy that combines unit resolution and set of support is used. Unit resolution requires all binary inferences (those requiring two parents) to have at least one of the parents be a unit clause. Set of support is a selection strategy for choosing parent clauses which requires at least one of the clauses to be from the set of support S which is defined as follows:

1) S is a subset of the initial clause set (can be arbitrarily chosen)
2) An inferred clause having a parent in S is also in S
3) no other clauses are in S.

See [Nilsson 82] for a more detailed exposition of these and other common selection strategies. Typically, all of the unit clauses will be included in the initial set of support, as well as the goal clause. Using unit resolution complements the restriction requiring sets of inferred clauses to be only unit clauses, as each of these new unit clauses can be used as a parent in a new inference.

Additionally, two types of subsumption are used. A clause C subsumes clause D if there exists a substitution Θ such that $C\Theta$ is a subset of D. Thus, P(x) subsumes P(a) $\vee$ Q(y) with $\Theta = \{a/x\}$, and R(a, z) $\vee$ P(w) subsumes R(a, y) $\vee$ P(c) $\vee$ Q(w) with $\Theta = \{y/z, c/w\}$. When a clause is subsumed it is deleted from the data base. The first type of subsumption that is used eliminates any clause subsumed by a unit clause. Thus, whenever a new unit clause is added, the data base is searched for all clauses which will be subsumed by this clause, with those found being removed. Also, when a non-unit clause has been inferred, the data base is searched for a unit clause that subsumes it. This search is optimized by the use of a data structure for the data base of clauses that has, for each predicate symbol P (and its negation), a list of pointers to exactly those clauses which contain P (or its negation). If such a clause is found, the new clause is not added. Since the time complexity of searching for general non-unit subsumption is exponential in the number of literals in the clauses, it is not performed. However, there are particular non-unit subsumptions that are quite powerful and which can efficiently be checked that are employed. This involves eliminating the non-unit parent in an inference when the child subsumes it. This will always be the case if the unit parent is more general than the literal that it matches in the non-unit parent, thus requiring only a single constant-time check. For example, if we resolve

~P(x) with
P(a) $\vee$ Q(a,y) $\vee$ R(c)

then the child

Q(a,y) $\vee$ R(c)

will subsume its non-unit parent. If however, we had

~P(a) resolving with
P(x) $\vee$ Q(x,y) $\vee$ R(c)

then the child

Q(a,y) $\vee$ R(c)

will not subsume the non-unit parent since there does not exist a substitution Θ that meets the conditions specified above. .

The complete set of rules that are implemented are below, with informal proofs of their correctness. The first three deal with intra-clausal matching of atoms (the irrelevant X-bundles of each X-clause were left off for notational clarity). The rule for matching complementary atoms in different X-bundles was entirely eliminated, since it always produces a linear number of non-unit children.

1) $X(A, \sim A, \underline{\mathbf{B}}_k) \vdash \{\sim B_i \mid 1 \leq i \leq k\}$
Since one of A or ~A must be true, all of the B_i's must be false.

2) $X(A, A, \underline{\mathbf{B}}) \vdash X(\underline{\mathbf{B}})$ and $\sim A$
In this case, since only one of the literals within the bundle can be true, A must be false, and the true literal must be one of the B's.

3) $A \vee X(A, \underline{\mathbf{B}}) \vdash A \vee X(\underline{\mathbf{B}})$
If A is true, the first A establishes the truth of the clause independent of the following X-bundle. If A is false, then one of the B's must be true.

The next two rules are special cases of X-resolution, and the third is a restriction on exclusion. See appendix one for the proofs of correctness.
4a) $X(\sim A, \underline{\mathbf{B}}),\ A \vdash X(\underline{\mathbf{B}})$
4b) $\sim A \vee \alpha,\ A \vdash \alpha$
5) $X(A, \underline{\mathbf{B}}_k),\ A \vdash \{\sim B_i \mid 1 \leq i \leq k\}$

5. Implementation

This set of rules and strategy proved to be useful within several domains. Using only the set of general inference rules and strategy outlined above, with no domain specific rules such as weighting of certain clauses or constants, there were several proofs that made only a few more inferences than the optimal proof tree. Figure 2 gives the proof tree that the theorem prover generated for a simple problem. All of the unit clauses were initially placed in the set of support, and the boldfaced clauses are subsumed by following clauses. Only one superfluous inference was made. Had we used traditional theorem proving methods, even writing down the clause set in CNF would have required 26 separate clauses, with the search space of unit resolutions numbering in the hundreds.

Another example was run using a commonsense rule base with 50 X-clauses, that contained an axiomatization of a highly partitioned domain, with such axioms as
PlaysGolf(z) ⊃ X(Pensioner(z), ExMovieStar(z))
Building(z) ⊃ X(FastFoodJoint(z), Factory(z), Jail(z)).
In typical proofs requiring 8-12 separate inferences, only 3 to 5 times this number of clauses was generated. Writing such a database for a clausal theorem prover would require 115 clauses, with the same proofs requiring at least 10-15 separate inferences. This space savings combined with shorter proof depth is significant, when one considers that in general, search spaces for proofs grow exponentially, thus raising the hope that more proofs will be made tractable using this method.

6. Conclusion

A logical form with corresponding inference rules has been presented that is useful in representing and reasoning within partitioned domains. Not only are declarative data bases smaller to write using the methods presented here, but fewer deductions are typically needed for a proof than using standard resolution. Due to the simplicity of its clause form and inference rules and its similarity to resolution-type systems, this extension can be added to many existing theorem provers with a minimum of effort, enabling better reasoning capabilities within taxonomic domains.

Special thanks to Pat Hayes, whose wit, insights and criticism were invaluable, and Leo Hartman, who never seemed to lose curiosity nor tire of my long-winded explanations.

Appendix 1

In the inference rules (X-rules) below, what is on the left hand side of the turnstile ($\vdash$) is what is given and will be called the parents, and what is on the right is the inference that can be made, and will be called the child. Since the X-bundles embed much information, several of the inferences result in sets of X-bundles being inferred. The first four X-rules have only a single parent. The rules are expressed in the propositional case, but hold with X-clauses in prenix form such that two identical propositional letters will be interpreted as two literals that unify.

X-RULES

1) $X(A, \sim A, \underline{\mathbf{B}}_k) \vee \alpha \quad \vdash \quad \{\sim B_i \vee \alpha \mid 1 \leq i \leq k\} \cup \{\sim B_i \vee \sim B_j \vee \alpha \mid 1 \leq i \leq k \;\&\; 1 \leq j \leq n\}$

2) $X(A, A, \underline{\mathbf{B}}_k) \vee \alpha \quad \vdash \quad A \vee X(\underline{\mathbf{B}}_k) \vee \alpha$ and $\sim A \vee \alpha$

3) $X(B, \underline{\mathbf{C}}_k) \vee x1(B, \underline{\mathbf{D}}_n) \vee \alpha$
$\vdash \quad \{B \vee X(\underline{\mathbf{C}}_k) \vee X(\underline{\mathbf{C}}_k) \vee \alpha\} \cup \{\sim B \vee \sim C_i \vee \sim D_j \vee \alpha \mid 1 \leq i \leq k \;\&\; 1 \leq j \leq n\}$

4) $X(B, \underline{\mathbf{C}}_k) \vee X(\sim B, \underline{\mathbf{D}}_n) \vee \alpha$
$\vdash \quad \{\sim B \vee \sim C_i \vee X(\underline{\mathbf{D}}_n) \vee \alpha \mid 1 \leq i \leq k\} \cup \{B \vee X(\underline{\mathbf{D}}_n) \vee \sim D_j \vee \alpha \mid 1 \leq j \leq n\} \cup \{\sim C_i \vee \sim C_j \vee \sim C_o \vee \sim D_l \vee \sim D_m \vee \sim D_p \vee \alpha \mid 1 \leq i < j < o < k \;\&\; 1 < l < m < p < n\}$

In proving the soundness of the above inference rules, we can show in each case that any model for a clause set that includes the parent must also be a model for the clause set with the addition of the child. I will prove the results for propositions, where the case for predicate formulas is similar save there must exist appropriate unifications.

Rule 1
If some X-bundle of α is true in our model (M) of the parent, then this same X-bundle will be true in each member of the inferred set, and thus M will be a model for each inferred X-clause. Let us take the case where no X-bundle of α from the parent is true in M. Since one of A and ~A must hold, and exactly one of A, ~A, and the B_i's must hold, we know that all of the B_i's must then be false and since at least one complemented B_i is in each child, all children must be true in M.

Rule 2
The case where some X-bundle in α in the parent is true is identical to Rule 1. Let

us take the case where all X-bundles in α are false. This says that exactly one of A, A, and the B_i's are true. If A is true, then both occurences of A are true, which renders the X-clause inconsistent. Thus, any model of the parent where α is false must have A false, which leaves exactly one of the B_i's true.

Case 3
If α is true in any model M of the parent (that is, some X-bundle in α is true in M), then M will also be a model for the child. If α is false, then we have two cases - when B is true, and when B is false. If B is false, then one or the other or both of the C and D X-bundles is true, which proves M to be a model for the first inferred set. If B is true, then either or both of all of the C_i's or D_j's are false so that M must be a model for the second set.

Case 4
If α is true in any model M of the parent, then M will also be a model for the children, since α is in all children. If α is false, then we again have two cases depending upon the truth value of B. If B is true, then either or both of all of the C_i's are false or exactly one of the D_j's is true which makes M a model for the first set. Analogously if B is false, which makes M a model for the second set. If α is false, then not more than one C_i or D_j can be true, which makes M also a model for the third inferred set.

The following two X-rules involve two parents. The first of these two is called the resolution rule, with the child called the resolvent, due to its similarity to Robinson's[3] resolution.

5) $X(B, \underline{C}_k) \vee \alpha_1 , \quad X(\sim B, \underline{E}_l) \vee \alpha_2 \ \vdash$
$X(\underline{C}_k, \underline{E}_l) \vee \alpha_1 \vee \alpha_2$

Case 5
Let M be some model of the clause set that includes the two parent clauses, and let the derived clause be called the resolvent. If either α is true in M, then the resolvent will likewise be true in M, since both α's are in this clause. If neither are true, then both of the X-bundles must be true in M. If B is true in M, then all of the C_i's are false, and since ~B is false, exactly one of the E_j's must be true. The case if B is false is symmetric. Thus, exactly one of the C_i's and E_j's is true, and M is a model for the resolvent. Note that if both k and l are 0 (the subscripts to $\underline{C}$ and $\underline{E}$ above), then this rule reduces to binary resolution:

$$X(B) \vee \alpha_1 , \quad X(\sim B) \vee \alpha_2 \ \vdash \alpha_1 \vee \alpha_2$$

The following is what I will refer to as the "exclusion" rule of inference, with any member of the child sets called an excluvent.

6) $$X(B, \underline{C}_k) \vee \alpha_1 \text{ , } X(B, \underline{E}_l) \vee \alpha_2 \vdash \\ \{\sim E_i \vee X(\underline{C}_k) \vee \alpha_1 \vee \alpha_2 \mid 1 \leq i \leq l\} \cup \\ \{\sim C_j \vee X(\underline{E}_l) \vee \alpha_1 \vee \alpha_2 \mid 1 \leq j \leq k\}$$

Case 6

Let M be a model of the set that includes the parent set. Let the inferred clauses be called "excluvents". As before, if either α is true in M, then it will also be true in each of the excluvents, and M will be a model for the excluvents. Therefore, in the following, let us assume that both α's are false, making both X-bundles true. If B is true in M, then all of the C_i's and E_j's must be false, thus making every excluvent true since its first literal will be true, hence M will be a model for the excluvents in this case as well. If B is false, then exactly one of the C_i's will be true, and exactly one of the E_j's will be true. Thus, M must be a model for all excluvents, since each excluvent has in it an X-bundle for either the C_i's or E_j's. Note that application of this rule results in an amount of clauses that are linear in the size of the X-bundles. Additionally, this rule results in no inferences when all X-bundles in both clauses are of degree one, since both k and l from the child set above are zero, and thus will never be applied in this case.

Appendix 2

In order to facilitate the following proof of completeness, I will introduce a few terms. An Exclusive Resolution refutation (XR-Proof) will be a derivation of the null clause from a set in Exclusive Normal Form from the repeated application of the X-rules given in the last section. A General Resolution refutation (GR-Proof) will be a refutation proof of a set in Conjunctive Normal Form using the inference rules of factoring and binary resolution, as well as tautology elimination, a semi-decidable procedure we know to be refutation complete for the first order predicate calculus. These inference rules are briefly the following. From the clause $A \vee A \vee \beta$ (where the β_i are some arbitrary number of disjuncted literals) we can infer by the factoring rule, $A \vee \beta$. From the clauses $A \vee \beta_1$ and $\sim A \vee \beta_2$ we can infer by the resolution rule, $\beta_1 \vee \beta_2$. From the clause $A \vee \sim A \vee \beta$ we can infer "True" by tautology elimination, and thus eliminate this clause from our clause set without affecting the completeness of our procedure.

An Expansion-Set of an X-clause Y is the minimal set of clauses in CNF that is truth functionally equivalent. For example,

$$\text{Expansion-Set}(X(A,B,C)) = \{(A \vee B \vee C), (\sim A \vee \sim B), (\sim B \vee \sim C), (\sim A \vee \sim C)\}.$$

Y is called the Reduce-clause of each of the clauses in its Expansion Set. Note that the Reduce-clause of a general clause is non-unique. The Expansion-Universe of a set of X-clauses in XNF is the union of the Expansion-Sets of the individual X-clauses.

Formally, the Expansion-Set of an X-bundle $X(\underline{\mathbf{B}}_n)$ is defined as follows:

$$\text{Expansion-Set}(X(\underline{\mathbf{B}}_n)) = \{B_1 \vee B_2 \vee \ldots \vee B_n\} \cup \{\sim B_i \vee \sim B_j \mid 1 \leq i < j \leq n\},$$

where $X(\underline{\mathbf{B}}_n)$ is an X-bundle of degree *n* (recall that an X-bundle of degree 1 is a literal). By this definition, the Expansion-Set of the null clause is the null clause. We can get the Expansion-Set of general X-clauses simply by replacing each X-clause with its Expansion-Set and putting the resulting expression into clausal form. In doing this, we will find that the Expansion-Set of a general X-Clause α with k X-bundles $E_1 \vee \ldots \vee E_k$ within it is

$$\text{Expansion-Set}(\alpha) = \{e_1 \vee e_2 \vee \ldots \vee e_k \mid e_i \in \text{Expansion-Set}(E_i)\}$$

If S is a clause set in XNF, with $S = \{ S_i \mid S_i \text{ is an X-clause} \in S \}$, then

$$\text{Expansion-Universe}(S) = \bigcup \text{Expansion-Set}(S_i).$$

We can now state the following *Equivalence* Lemma.

Equivalence Lemma:

A set of clauses S in XNF is unsatisfiable iff Expansion-Universe(S) is unsatisfiable.

Proof:

This follows immediately from the definitions of Expansion-Set of an X-bundle and X-clause, and the definition of Expansion-Universe, since all are given as truth-functional equivalences to sentences in first order logic with the standard connectives.

We are now ready to prove the following theorem.

Theorem:

Exclusive Resolution is refutation complete.

Proof:

We will prove this by the following. Given an XNF set S, there exists an XR-proof iff there exists a GR-proof of the Expansion-Universe(S).

Only If (non-constructive)

Let us assume that there exists an XR-proof of S. We have shown that all X-rules are sound. Therefore, S must be unsatisfiable, since we were able to derive the null clause through application of the X-rules. The Expansion-Universe(S) is likewise unsatisfiable, by the Equivalence Lemma. Since we know that GR is complete, we know that any unsatisfiable set will be proved to be such by GR. Therefore, there exists a GR-proof of the Expansion-Universe(S).

If (constructive)

Let us assume that there exists a GR-proof of the Expansion-Universe(S). We can view this refutation as a tree, with nodes being labeled by clauses, and arcs running only from each of the parents to the resolvent clause. The root of this tree is the null clause, and the leaves are the members of the Expansion-Universe(S). Central to this proof are the following lemmas. These lemmas state formally that for every inference in the GR-Proof, there exists a corresponding inference in the XR-Proof. This guarantees that no information is lost using the XR-proof methods.

Lemma:

```
1)      For all clauses p1, c, and all X-clauses p1'
          If p1 is in the Expansion-Set of p1', and
          c is a child of a factoring inference applied to p1 then
        There exists an X-rule R, and an X-clause c' such that
          c' is a child of p1' upon the application of rule R, and
          c is in the Expansion-Set of c'.
```

2) For all clauses p1, p2, c, and all X-clauses p1', p2',
 If p1 is in the Expansion-Set of p1', and
 p2 is in the Expansion-Set of p2', and
 c is a child of a resolution applied to p1 and p2, then
 There exists an X-rule R, and an X-clause c' such that
 c' is a child of p1' and p2' upon the application of
 rule R, and c is in the Expansion-Set of c'.

Proof of Lemma (1):

Consider a factoring inference in the GR-proof. It has the form

$$A \vee A \vee \gamma \models A \vee \gamma$$

where γ represents some arbitrary disjunct of literals, and A can be either an atom or the negation of an atom. These two clauses will be called p1 and c respectively in the following proofs, with the corresponding X-clauses called p1' and c'. We must look at every possible X-clause (p1') such that the clause $A \vee A \vee \gamma$ is in its Expansion-Set, and demonstrate the existence of an X-rule such that some child that is inferred (c') from p1' and this X-rule has $A \vee \gamma$ within its Expansion-Set.

We need consider only the general cases when the A's in the Reduce-Clause of $A \vee A \vee \gamma$ are from the same X-bundle, and when they are from different X-bundles. Note that in some of the proofs below, there might be an expression of the form

$$\text{Expansion-Set}(\alpha) \vee \text{Expansion-Set}(\beta) \vee \text{Expansion-Set}(\xi).$$

This will be shorthand for the set of X-clauses

$$\{a \vee b \vee c \mid a \in \text{Expansion-Set}(\alpha),\ b \in \text{Expansion-Set}(\beta), \text{ and } c \in \text{Expansion-Set}(\xi)\}.$$

Also, a boldfaced propositional letter will sometimes be given without the underline. This will represent a disjunction of literals. Since $\{B_1 \vee B_2 \vee \dots \vee B_n\} \in \text{Expansion-Set}(X(\underline{\mathbf{B}}_n))$, $\mathbf{B}_n$ will be used as a shorthand for $\{B_1 \vee B_2 \vee \dots \vee B_n\}$.

Case 1) Both A's are in the same X-bundle. We must consider this for the cases when the A's are both positive, both negative, and one positive and the other negative in the X-clause. The form of the X-clause in the positive case will be

$$X(A, A, \underline{\mathbf{B}}_k) \vee \boldsymbol{\alpha}.$$

The inference is X-rule (2) and the child clause is

$$A \vee X(\underline{\mathbf{B}}_k) \vee \boldsymbol{\alpha}.$$

Since $A \vee A \vee \gamma \in \text{Expansion-Set}(X(A, A, \underline{\mathbf{B}}_k) \vee \alpha)$, (our initial assumption), this means that $\gamma \in \mathbf{B}_k \vee \text{Expansion-Set}(\alpha)$ in the positive case, since all B's are positive in any clause in the Expansion-Set in which both A's are positive. Thus, $A \vee \gamma \in \text{Expansion-Set}(A \vee X(\underline{\mathbf{B}}_k) \vee \alpha)$, since

$$\text{Expansion-Set}(A \vee X(\underline{\mathbf{B}}_k) \vee \alpha) =$$
$$\text{Expansion-Set}(A) \vee \text{Expansion-Set}(X(\underline{\mathbf{B}}_k)) \vee \text{Expanion-Set}(\alpha) \supseteq$$
$$A \vee \mathbf{B}_k \vee \text{Expansion-Set}(\alpha) \ni A \vee \gamma.$$

Now let us consider when A is negative. The form of the X-clause will be

$X(\sim A, \sim A, \underline{B}) \vee \alpha$.

The inference is X-rule (2) and the child clause is

$A \vee \alpha$.

Note that

$\text{Expansion-Set}(X(\sim A, \sim A, \underline{B})) \vee \text{Expansion-Set}(\alpha) \ni$
$(A \vee A) \vee (\gamma)$, and thus, $A \vee \gamma \in \text{Expansion-Set}(A \vee \alpha)$.

In the case when the A's are complementary in the X-clause, there does not exist a clause in its Expansion-Set that would give rise to a factoring instance. Thus this case is irrelevant.

Case 2) The A's are in different X-bundles.
We must again consider when the A's are complementary, when they are both positive, and both negative. In the first case, the form of the X-clause is

$X(A, \underline{C}_k) \vee X(\sim A, \underline{D}_n) \vee \alpha$.

The X-rule is (4), and the child is an element of

$\{A \vee X(\underline{C}_k) \vee \sim D_j \vee \alpha \mid 1 \leq j \leq n\}$.

In order for $p1 \in \text{Expansion-Set}(p1')$,

$\text{Expansion-Set}(X(A, \underline{C}_k) \vee X(\sim A, \underline{D}_n) \vee \alpha) \supseteq$
$(A \vee C_k) \vee (A \vee \sim D_j) \vee \text{Expansion-Set}(\alpha) \ni$
$A \vee A \vee \gamma$.

Thus, $\gamma \in C_k \vee \sim D_j \vee \text{Expansion-Set}(\alpha)$. Therefore,

$\text{Expansion-Set}(A \vee X(\underline{C}_k) \vee \sim D_j \vee \alpha) \supseteq$
$A \vee \text{Expansion-Set}(X(\underline{C}_k)) \vee \sim D_j \vee \text{Expansion-Set}(\alpha) \supseteq$
$A \vee C_k \vee \sim D_j \vee \text{Expansion-Set}(\alpha) \ni$
$A \vee \gamma$.

The case when the A in the first X-bundle is negative and the A in the second X-bundle is positive is symmetric.

In the positive case, the X-clause form is

$X(A, \underline{C}_k) \vee X(A, \underline{D}_n) \vee \alpha$,

the X-rule is (3) and the child is

$\{A \vee X(\underline{C}_k) \vee X(\underline{D}_n) \vee \alpha\}$.

There is only a single clause in Expansion-Set(p1') in which both A's are positive, and in this clause all C's and D's are also positive. Thus, $\gamma \in \text{Expansion-Set}(X(\underline{C}_k) \vee X(\underline{D}_n) \vee \alpha)$, and $c \in \text{Expansion-Set}(c')$.

When the A's are negative, the X-clause form is

$X(\sim A, \underline{C}_k) \vee X(\sim A, \underline{D}_n) \vee \alpha$,

the X-rule is (3) and the child is a member of

$\{A \vee \sim C_i \vee \sim D_j \vee \alpha \mid 1 \leq i \leq k \ \& \ 1 \leq j \leq n\}$.

Since both ~A's must be complemented in p1, all C's and D's are complemented in p1. By similar arguments to those in previous cases, we find $c \in \text{Expansion-Set}(c')$.

Since we have considered all cases pertaining to factoring, we have established Lemma (1).

Binary resolutions have the form

$$(A \vee \gamma_1) \, , \, (\sim A \vee \gamma_2) \;\vdash\; (\gamma_1 \vee \gamma_2).$$

We can look at the various cases of X-clauses that these clauses must be expanded from. I will call the two parent clauses p1 and p2, and the child c, with the corresponding X-clauses called p1', p2', and c'.

There are two general cases - first, when both parents, p1 and p2, are from the same Reduce-Clause, and second, when they are from different Reduce-Clauses.

Case 1) p1 and p2 are expanded from the same Reduce-Clause.
There are several sub-cases to explore, namely when the unified literal (A, in these examples) occurs in the same X-bundle, and when the A's occur in different X-bundles of the Reduce-Clause.

Case 1a) A occurs in the same X-bundle.
We must consider the cases when the A's have the same truth values, and when they have complementary truth values. When the truth values are the same, the form of the X-clause is

$$X(A, A, \underline{B}_k) \vee \alpha.$$

However, if we look at the clauses in the expansion set, those that contain A are all of the form

$$A \vee A \vee \beta \quad \text{or} \quad \sim A \vee \sim A \vee \Gamma.$$

Thus, any resolution between two such clauses will result in a child of the form

$$A \vee \sim A \vee \beta \vee \Gamma$$

which can then be eliminated due to the fact that it is a tautology. We need therefore not consider the case when the A's are either both positive or both negative, since the resulting tautological inferences will not occur in the GR-proofs.

In the case where the A's have complementary truth values, the form of the X-clause is

$$X(A, \sim A, \underline{B}_k) \vee \alpha,$$

the X-rule is rule (1), and the child has the form

$\sim B_i \vee \sim B_j \vee \alpha$, for some i and j.

Clearly all clauses in the Expansion-Set in which both A and ~A occur will be eliminated due to the tautologies, since for any clause in which two A's occur, one will be positive and the other negative. Thus, resolutions will only occur between those clauses in which only one of the A's are present. These will be of the form

$A \vee \sim B_i \vee \text{Expansion-Set}(\alpha)$ and $\sim A \vee \sim B_j \vee \text{Expansion-Set}(\alpha)$ which means that $\gamma_1 \in \sim B_i \vee \text{Expansion-Set}(\alpha)$ and $\gamma_2 \in \sim B_j \vee \text{Expansion-Set}(\alpha)$, for some i and j. Thus,

$\gamma_1 \vee \gamma_2 \in \text{Expansion-Set}(\sim B_i \vee \sim B_j \vee \alpha)$, and thus $c \in c'$.

Case 1b) The A's are expanded from different X-bundles.
There are two subcases here - when the A's have the same truth value, and when they are complementary. In the first case, the X-clause is of the form

$$X(B, \underline{C}_k) \vee X(B, \underline{D}_n) \vee \alpha.$$

Examining the Expansion-Set of this clause, we find that no member that we need to consider has both B and ~B in it. Thus, those clauses that we will consider have either no B's, one B, or two B's that are either both positive or both negative. An exhaustive examination of every possible resolution unearths the fact that every child can be eliminated due to a resulting tautology - thus, we need not further consider this case.

When the truth values are complementary, the X-clause form is

$$X(B, \underline{C}_k) \vee X(\sim B, \underline{D}_n) \vee \alpha,$$

the X-rule is (4) and the child is in the set

$$\{\sim C_i \vee \sim C_j \vee \sim C_o \vee \sim D_l \vee \sim D_m \vee \sim D_p \vee \alpha \mid 1 \le i \le j \le o \le k \ \& \ 1 \le l \le m \le p \le n\}.$$

Although this rule is far from intuitive, it results again from an exhaustive search of the Expansion set, considering all cases when a resolution is possible. In all cases where the child is not a tautology, it will be a member of the above set. That $c \in$ Expansion-Set(c') follows by similar arguments as in previous sections.

Case 2) p1 and p2 are expanded from different Reduce-Clauses.
We must take the case where both A's have the same truth value, and where both A's are of different truth values.

Case 2a) When they have the same truth value, the X-clause form is

$$X(A, \underline{C}_k) \vee \alpha_1 \quad , \quad X(A, \underline{E}_l) \vee \alpha_2,$$

the X-rule is (6), and the child is in the set

$$\{\sim E_i \vee X(\underline{C}_k) \vee \alpha_1 \vee \alpha_2 \mid 1 \le i \le l\} \cup$$
$$\{\sim C_j \vee X(\underline{E}_l) \vee \alpha_1 \vee \alpha_2 \mid 1 \le j \le k\}.$$

In order for the A's to be complementary in p1 and p2 either all C_i's will be positive and extant in c, and some E_i will be complemented or all E_i's will be positive and extant in c, and some C_i will be complemented. The two sets from which the child can be a member of covers both of these possibilities.

When the A's have different truth value, the X-clauses have the form

$$X(A, \underline{C}_k) \vee \alpha_1 \ , \quad X(\sim A, \underline{E}_l) \vee \alpha_2,$$

the X-rule is (5), and the child is

$$X(\underline{C}_k, \underline{E}_l) \vee \alpha_1 \vee \alpha_2.$$

In order that both A's are complemented in the Expansion-Sets, all resolutions will result in either all C_j's and E_i's positive, in which case these children will be in the Expansion-Set of the X-clause child, or one $\sim C_j$ and one $\sim E_i$ will be in the resolvent, which is also in the Expansion-Set of the X-clause child. To clarify, note that

$$\text{Expansion-Set}(X(A, \underline{C}_k)) \ni A \vee C_k \text{ and}$$

Expansion-Set(X($\sim$A, $\underline{E}_l$)) $\ni$ $\sim$A $\vee$ E_l. Additionally
Expansion-Set(X(A, $\underline{C}_k$)) $\ni$ $\sim$A $\vee$ $\sim C_j$ and
Expansion-Set(X($\sim$A, $\underline{E}_l$)) $\ni$ A $\vee$ $\sim E_i$.

Since we have proven exhaustively all cases, we have established the truth of both Lemmas 1) and 2). Thus by these Lemma's we know that the null clause that was derived from the GR-Proof is in the Expansion-Set of some X-clause in the XR-Proof. But by definition, this X-clause must also be the null clause. Therefore, there must exist an XR-Proof, completing the proof that the X-rules when applied to a set in XNF is refutation complete.

It seems clear that many of these rules are unduly complex, and even unnecessary. X-rule (1) is a prime example, since the set $\{\sim B_i \vee \sim B_j \vee \alpha \mid 1 \leq i \leq k \ \& \ 1 \leq j \leq n\}$ appears to be entirely subsumed by the set $\{\sim B_i \vee \alpha \mid 1 \leq i \leq k\}$. The problem stems from the fact that the central Lemmas of the completeness proof demand a one to one correspondence between steps in the respective proofs. But X-rules are actually more powerful than binary refutation in the sense that one X-rule inference may take several corresponding binary resolution inferences, which figures 1 and 2 demonstrate graphically. This work however, does establish one set of complete X-rules, and it appears to be a rather simple extension to develop a completeness proof of a more refined set of X-rules by showing their equivalence to this set, rather than showing their equivalence to general resolution (or natural deduction, or some other different but complete set of inferences). In fact the set of rules that are implemented (save for the unit-preference strategy) may in fact be complete, although to prove this, I believe some ordering strategy must be invoked which indefinitely defers the application of any X-rule that results in a non-singular X-clause.

REFERENCES

[1] Hayes, P.J. "Naive Physics I: Ontology for Liquids", Working Paper 63, Institut pour les Etudes Semantiques et Cognitives, Geneva, 1978.

[2] Hendrix, G.C. "Encoding Knowledge in Partioned Networks", in Associative Networks, ed. Findler, N.V. 1979.

[3] Nilsson, N. Principles of Artificial Intelligence, Tioga Publishing Company, Palo Alto, 1980.

[4] Robinson, J.A. "A Machine-Oriented Logic Based on the Resolution Principle" Journal of ACM 12 (1965) 23-41.

[5] Shapiro, S.C. "The SNePS Semantic Network Processing System", in Associative Networks, ed. Findler, N.V. 1979.

[6] Stickel, M.E. "A NonClausal Connection-Graph Resolution Theorem-Proving Program", Technical Note 268 SRI International, Menlo Park, CA Oct. 1982.

[1] $\neg A \vee \neg B_1$
[2] $\neg A \vee \times(D_1, D_2, D_3)$
[3] B_1
[4] B_n
[5] $\neg C$
[6] $A \vee C \vee \times(B_1, B_2, \ldots, B_n)$

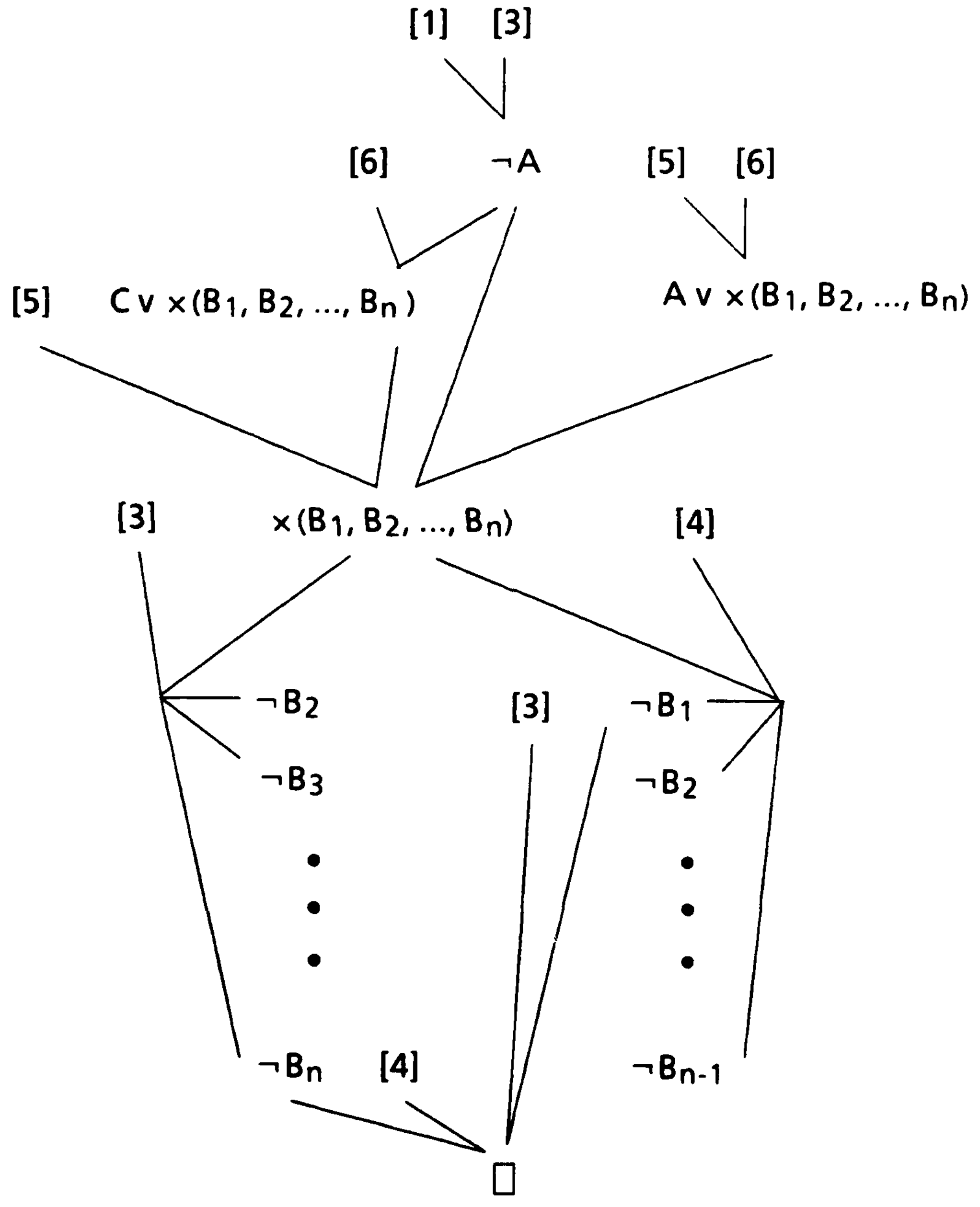


figure 1

Given:

[1] Animal(y) ⊃ X(Eeyore(y), Kanga(y), Tigger(y), Owl(y), Piglet(y), Pooh(y))

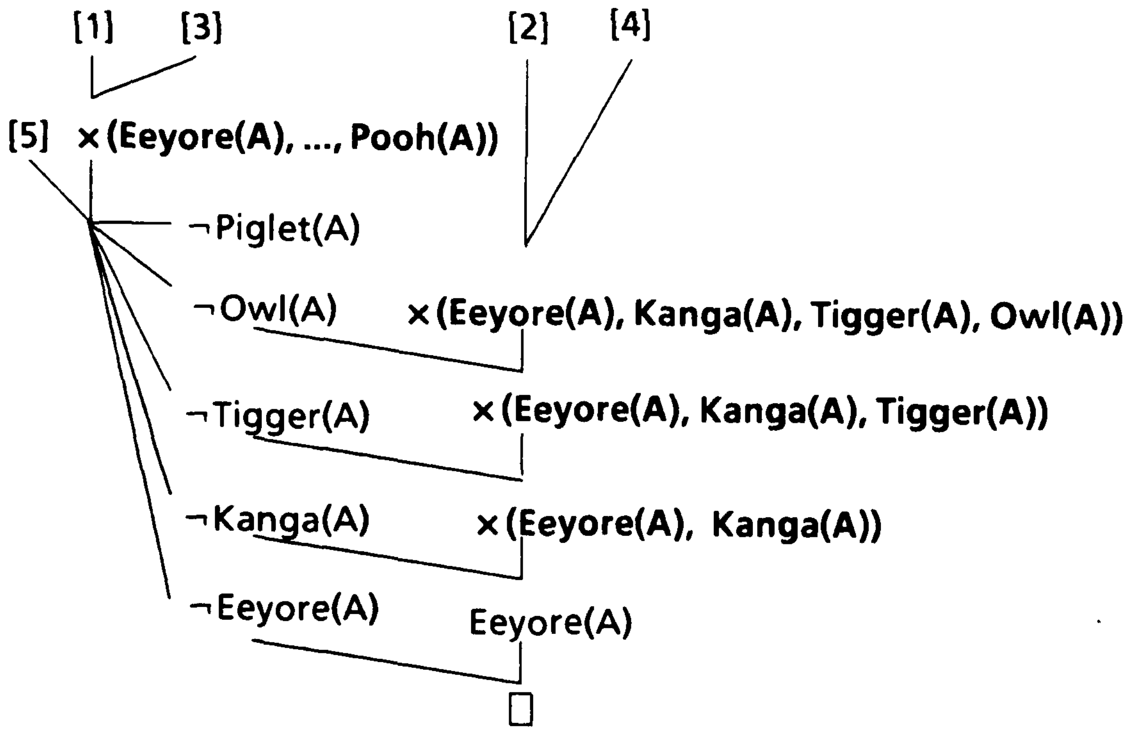
[2] ¬MainCharacter(y) ⊃ X(Eeyore(y), Kanga(y), Tigger(y), Owl(y))

[3] Animal(A)

[4] ¬MainCharacter(A)

Prove: ¬Pooh(A) *negate and add*

[5] Pooh(A)



[1]
[3]
[2]
[4]
[5]
× (Eeyore(A), ..., Pooh(A))
¬Piglet(A)
¬Owl(A)
× (Eeyore(A), Kanga(A), Tigger(A), Owl(A))
¬Tigger(A)
× (Eeyore(A), Kanga(A), Tigger(A))
¬Kanga(A)
× (Eeyore(A), Kanga(A))
¬Eeyore(A)
Eeyore(A)
□

figure 2

END

DTIC

5-86